УДК 658:512.011:681.326:519.713

BAGHDADI AMMAR AWNI ABBAS (БАГХДАДИ АММАР АВНИ АББАС),
В.И. ХАХАНОВ, Е.И. ЛИТВИНОВА

МЕТОДЫ АНАЛИЗА И ДИАГНОСТИРОВАНИЯ ЦИФРОВЫХ УСТРОЙСТВ (АНАЛИТИЧЕСКИЙ ОБЗОР)

Рассматриваются основные направления технологий проектирования, валидации и обеспечения качества (quality assurance) систем на кристаллах и в пакетах кристаллов.

Введение

Цель – аналитический обзор квантовых методов вычислений, технологий тестирования и диагностирования цифровых систем на кристаллах при их проектировании и верификации, ориентированный на повышение быстродействия программных и аппаратных средств анализа цифровых устройств, а также существенное увеличение выхода годной продукции и уменьшение времени ее выхода на рынок микроэлектроники.

Задачи: 1) Особенности классических и квантовых вычислений. 2) Построение моделей цифровых систем. 3) Моделирование исправного поведения цифровой систем. 4) Моделирование неисправностей цифровой системы. 5) Генерация тестов. 6) Методы диагностирования неисправностей. 7) Построение кубитных моделей цифровых систем.

Источники: классические и квантовые вычисления [1-20]; квантовые вентили [6, 15, 27]; виды неисправностей цифровых схем [28-32, 36, 40-44]; генерация тестов [36, 37, 45]; моделирование цифровых систем [28-35, 38, 39]; технологии тестирования и диагностирования систем на кристаллах [28-34, 36-39].

1. Классический и квантовый бит информации

Классический и квантовый компьютеры оперируют числами. Двоичная система счисления классического компьютера основывается на использовании двух чисел: 0 и 1. Количество информации, хранящееся в одной двоичной ячейке, равно 1 биту. В двух двоичных ячейках хранится 2 бита информации, которые могут составлять следующие комбинации: 00, 01, 10, 11. В регистре, состоящем из n двоичных ячеек, хранится n бит информации, количество комбинаций возможных значений которых равно 2^n.

В квантовом компьютере [1-20] информация также может быть представлена в виде двоичных чисел. Однако в ячейке, называемой квантовым битом (кубитом), может храниться не одна из двух цифр двоичного счисления (0 или 1), а одновременно обе эти цифры [1,2]. Например, в двух кубитах могут храниться одновременно 4 двоичных числа 00, 01, 10, 11. В n кубитах могут храниться одновременно 2^n двоичных чисел длины n. Кубиты могут быть связаны друг с другом (запутаны): изменение одного из них приводит к изменениям других. Совокупность запутанных кубитов представляет собой заполненный квантовый регистр, который может не только находиться во всевозможных комбинациях составляющих его битов, но и реализовывать всевозможные зависимости между ними [2]. В процессе вычислений на квантовом компьютере одновременно обрабатываются все биты квантового регистра (имеет место квантовый параллелизм).

В процессе вычислений квантовые компьютеры выполняют логические операции над квантовыми состояниями путём унитарных преобразований, не нарушающих квантовые суперпозиции, в соответствии с алгоритмом [3]:

1) запись (подготовка, инициализация) начального состояния;

2) вычисление (унитарные преобразования начальных состояний);

3) вывод результата (измерение, проецирование конечного состояния).

Классический компьютер оперирует битами – булевыми переменными, принимающими значения 0 и 1 – и на любом этапе вычислений в каждом бите хранятся определенные значения, используемые для вычислений. На первом этапе вычислений в каждый бит записываются исходные данные –определенные значения (0 или 1).

Квантовый компьютер оперирует состояниями квантового регистра. Данные в процессе вычислений представляют собой квантовую информацию, которая по окончании вычислительного процесса преобразуется в классическую путём измерения конечного состояния квантового регистра. Выигрыш по быстродействию в квантовых алгоритмах достигается за счёт того, что при применении одной квантовой операции большое количество коэффициентов суперпозиции квантовых состояний, которые в виртуальной форме содержат классическую информацию, преобразуется одновременно (квантовый параллелизм).

2 Квантовые вентили

2.1. *Простые квантовые вентили.* По аналогии с обычным электронным цифровым компьютером вентиль квантовой вычислительной системы может обрабатывать один или несколько кубитов [6, 15]. Квантовая система имеет входные и выходные состояния, определяемые (задаваемые) одним или несколькими кубитами. Система преобразует их в соответствии с аксиомами квантовой механики. Состояния квантовой системы преобразуются с помощью унитарных, обратимых (инвертирующихся) операторов. Следовательно, квантовые вентили должны иметь одинаковое количество входных и выходных кубитов.

Простой квантовый вентиль A, преобразующий один входной кубит в один выходной, представлен на рис. 1.

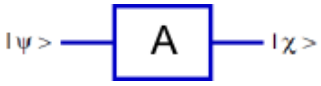

Рис. 1. Квантовый вентиль

Здесь предполагается, что в квантовом вентиле поток информации направлен слева направо, поэтому стрелки на рис. 1 не показаны. Данное упрощение графического представления квантового вентиля стало возможным благодаря тому факту, что квантовые вентили преобразуют кубиты с помощью унитарных обратимых операторов. В этом случае графическое представление логических вентилей, преобразующих классические биты в электронном цифровом компьютере, не является удобным для квантовых вентилей.

На рис. 1 $|\psi>, |\chi> \in C^2$ – кубиты; A: $C^2 \longrightarrow C^2$ – унитарный линейный обратимый оператор. Удобно использовать матричное представление квантовых вентилей:

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix},$$

где для кубитов $|\psi> = \alpha|0> + \beta|1>, |\chi> = \gamma|0> + \delta|1>$, для которых $A|\psi> = |\chi>$, справедливо:

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}\begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \gamma \\ \delta \end{pmatrix}. \qquad (1)$$

Пример квантового вентиля NOT, задаваемого унитарной матрицей:

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

Нетрудно заметить, что в этом случае выражение (1) преобразуется к виду:

$$X(\alpha|0> + \beta|1>) = \beta|0> + \alpha|1>.$$

Другими словами, вентиль NOT представляет собой переключатель между состояниями $|0>$ и $|1>$.

Другой пример квантовых вентилей, описываемых унитарными матрицами:

$$Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \; Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix},$$

которые действуют на данный кубит в соотвествии с выражениями:

$$Y(\alpha|0> + \beta|1>) = -i(-\beta|0> + \alpha|1>),$$

$$Z(\alpha|0> + \beta|1>) = \alpha|0> - \beta|1>.$$

X, Y и Z называют матрицами Паули.

Вентиль Адамара осуществляет самообратимую операцию формирования суперпозиции состояний и определяется унитарной матрицей [6, 15]:

$$H = (1/\sqrt{2})\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}.$$

Справедливо следующее выражение

$$X^2 = Y^2 = Z^2 = H^2 = I,$$

которое означает, что каждый вентиль X,Y,Z и H определяется с помощью корня квадратного единичной матрицы и соответствующего квантового вентиля I.

2.2. *Многокубитные квантовые вентили.* В многокубитном вентиле должно быть одинаковое количество кубитов на входе и выходе [6].

Двухкубитные вентили соответствуют операциям поворота в гильбертовом пространстве двух взаимодействующих кубитов, которые не могут быть представлены в виде прямого произведения независимых однокубитовых операций [15].

Основным двухкубитовым вентилем является обратимый контролируемый инвертор или оператор «контролируемое НЕ» (CNOT) с двумя входными и двумя выходными кубитами (рис. 2), который функционирует в соответствии с выражениями:

$$|00> \mapsto |00>, \quad |01> \mapsto |01>,$$
$$|10> \mapsto |11>, \quad |11> \mapsto |10>.$$

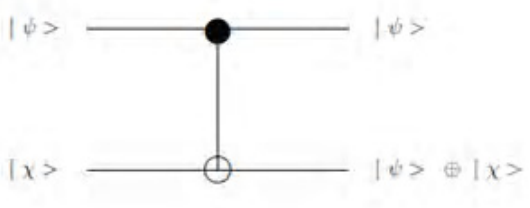

Рис. 2. Контролируемое НЕ

Когда $|\psi> = |0>$, то $|\psi> \oplus |\chi> = |\chi>$; для $|\psi> = |1>$ справедливо $|\psi> \oplus |\chi> = X|\chi>$.

Вентиль CNOT описывается матрицей:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}\begin{pmatrix} \alpha \\ \beta \\ \gamma \\ \delta \end{pmatrix} = \begin{pmatrix} \alpha \\ \beta \\ \delta \\ \gamma \end{pmatrix},$$

где $|\psi> = \alpha|0> + \beta|1>, |\chi> = \gamma|0> + \delta|1>$.

Кубит $|\psi>$ является контролирующим, кубит $|\chi>$ – контролируемым, над которым производится операция NOT при условии, что первый кубит находится в состоянии $|1>$.

Двухкубитовый оператор обмена состояниями кубитов SWAP может быть реализован путем последовательного выполнения трех операций CNOT (рис. 3) [15] и описывается матрицей:

$$SWAP = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

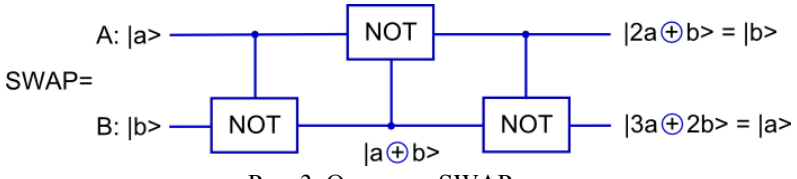

Рис. 3. Оператор SWAP

Трехкубитовый вентиль Тоффоли (CCNOT) представлен на рис. 4 и содержит два управляющих кубита A и B, а также один управляемый C.

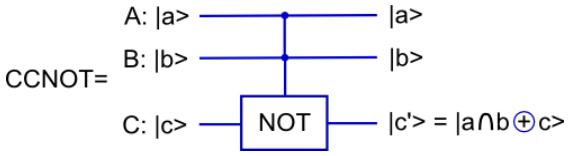

Рис. 4. Вентиль Тоффоли

Вентиль Тоффоли описывается матрицей 8x8 в базисных состояниях |0,0,0>, |0,0,1>, |0,1,0>, |1,0,0>, |0,1,1>, |1,0,1>, |1,1,0>, |1,1,1>:

$$CCNOT = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Данная операция может быть также реализована в виде пяти двухкубитовых операций. N-битный обобщенный вентиль Тоффоли описывается как $(k_1,k_2,...,k_n) \to (k_1,k_2,...,(k_1,k_2,...,k_{n-1}) \oplus k_n)$ [20]. Вентиль NOT является частным случаем вентиля Тоффоли, для которого n=1, вентиль CNOT –частным случаем, когда n=2.

Расширенный n+1-битный вентиль Тоффоли (extended Toffoli gate, ETG) с двумя управляемыми линиями (o_n, o_{n+1}) показан на рис. 5. ETG имеет входной вектор $(k_1,k_2,...,k_n,k_{n+1})$ и выходной вектор $(o_1,o_2,...,o_n,o_{n+1})$, где $o_j = k_j$ для $j = \overline{1,(n-1)}$, $o_n = k_1,k_2,...,k_{n-1} \oplus k_n$, $o_{n+1} = k_1,k_2,...,k_{n-1} \oplus k_{n+1}$. Первые n-1 битов являются управляющими, последние два бита – управляемыми.

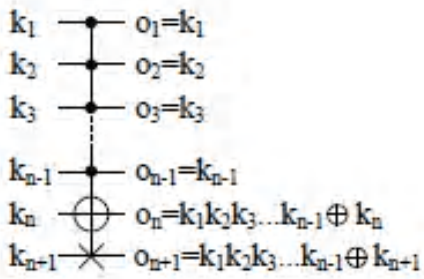

Рис. 5. n+1-битный вентиль Тоффоли

3. Виды дефектов

Подходы к моделированию неисправностей, диагностированию, тестированию и обеспечению отказоустойчивости цифровых схем имеют важное значение для создания надежных изделий [40]. С развитием VLSI технологии существенно увеличилось количество компонентов, размещаемых на одном кристалле, и, как результат – возросла степень интеграции и количество неисправностей.

В процессе проектирования цифрового изделия необходимо периодически проверять (путем тестирования) корректность функционирования схемы и отсутствие неисправностей, а также обеспечить корректную работу схемы в случае появления неисправностей (отказоустойчивость) [36].

Под неисправностью схемы понимают физический дефект одного или более компонентов [28, 29, 32]. Различают постоянные и нерегулярные неисправности. Постоянные (жесткие) неисправности могут быть следствием разрушения или износа компонента. Нерегулярные (мягкие) дефекты проявляются в определенные промежутки времени и могут быть кратковременными (transient) и перемежающимися (intermittent) [40-44].

Неисправности могут быть логическими и параметрическими. Логическая неисправность изменяет булеву функцию, которая реализуется схемой. Параметрическая – изменяет значение параметра схемы (ток, напряжение). К параметрическим дефектам относят неисправности задержки, связанные с различным временем прохождения сигнала через логические вентили, что приводит к гонкам (состязаниям) сигналов [46].

Моделирование большого количества физических дефектов может быть основано на использовании одной модели логической неисправности, что позволяет существенно уменьшить сложность моделирования. Модель логической неисправности не зависит от технологии имплементации проекта, а тесты, разработанные для обнаружения логической неисправности, могут применяться также и для выявления физических дефектов.

Модель логической неисправности может быть явной или неявной. Явная модель неисправности определяет пространство неисправностей, в котором каждая неисправность может быть идентифицирована, а неисправности, подлежащие анализу, могут быть явно описаны. Явная модель неисправности практически применима, если ее размерность не является слишком большой. Неявная модель описывает пространство неисправностей путем совокупной идентификации неисправностей определенного типа их характеристическими признаками. Моделирование неисправностей тесно связано с моделированием системы. Неисправности, определяемые в сочетании со структурной моделью, относятся к структурным неисправностям, проявляющимся в изменении структуры межсоединений компонентов. Функциональные неисправности определяются в сочетании с функциональной моделью. Например, функциональная неисправность может проявляться в изменении таблицы истинности компонента или искажении RTL операции.

Классы неисправностей. Существует три класса логических неисправностей: константная (stuck-at-fault), мостиковая (bridging fault) и неисправность задержки (delay fault).

Наиболее распространенная модель константной неисправности – одиночная константная неисправность. Сущность модели заключается в том, что неисправность логического вентиля приводит к «залипанию» логического 0 (константа 0, s-a-0) или логической 1 (константа 1, s-a-1) на одном из его входов или выходов [42, 44].

Модель константной неисправности также используется для представления кратных неисправностей в схеме. При этом предполагается, что более, чем на одной линии схемы имеется константная неисправность s-a-0 или s-a-1. Другими словами, совокупность константных неисправностей существует в схеме в одно и то же время. Разновидностью кратной неисправности является однонаправленная неисправность – когда все конституенты (составляющие части) неисправности представляют собой s-a-0 или s-a-1, но не обе одновременно.

Модель константной неисправности не является эффективной при моделировании СБИС (Very Large Scale Integrated, VLSI), построенных по CMOS технологии. Неисправности в CMOS схемах не обязательно представляют собой логические дефекты, которые могут быть описаны моделью константной неисправности. Дефекты CMOS-схем отображаются также моделями устойчивых обрывов транзисторов SOP (stuck-open) и устойчивых замыканий транзисторов SON (stuck-on) [42].

Мостиковые неисправности типа «короткое замыкание» представляют собой постоянные дефекты, которые не могут быть смоделированы константной неисправностью. Короткое замыкание возникает, когда две или более сигнальных линий схемы электрически связаны друг с другом. Мостиковые неисправности вентильного уровня классифицируются следующим образом: входные – вызваны коротким замыканием входов логического элемента, неисправности типа обратной связи – вызваны замыканием входной и выходной линий, а также неисправности без обратной связи, которые не относятся к первым двум типам. В теории моделирования мостиковых неисправностей делается предположение, что вероятность замыкания более двух линий является низкой и логика межсоединений реали-

зуется в виде связей. Мостиковая неисправность в положительной логике возникает в том случае, когда ее поведение описывается проводным AND (0 является доминантным логическим значением), и в отрицательной логике - когда ее поведение описывается проводным OR (1 является доминантным логическим значением).

Неисправности задержки [41, 43]. Небольшое количество дефектов, которые могут вызвать разрывы и короткие замыкания в схеме, имеют достаточно высокую вероятность появления из-за наличия отклонений параметров производственного процесса. Дефекты могут также приводить к нарушениям временных параметров схемы без изменения логики ее работы: задержка переключения сигнала из 0 в 1, и наоборот. Существует два вида неисправностей задержки: неисправность задержки вентиля и неисправность задержки пути. Задержка вентиля используется для моделирования дефектов, при которых время прохождения сигнала через вентиль превышает предельно-допустимое. Данная модель может быть использована только для изолированных, не транспортируемых дефектов, например, несколько малых задержек. Модель задержки пути может быть использована как для изолированных, так и для транспортируемых дефектов. При этом предполагается, что неисправность проявляется в случае, если задержка распространения сигнала вдоль линии схемы превышает допустимое значение.

Кратковременные и перемежающиеся неисправности рассматриваются как временные дефекты. Основная часть неисправностей цифровых схем вызвана именно временными дефектами, которые характеризуются сложностью выявления и устранения. Кратковременные дефекты являются неповторяющимися и вызываются, как правило, флуктуациями напряжения питания или воздействием радиационного излучения. Они являются основной причиной отказа элементов памяти систем на кристаллах.

Перемежающиеся неисправности могут появляться в результате нарушения межсоединений, применения дефектных компонентов, воздействия внешних факторов (температура, влажность, вибрация) или быть следствием ошибок проектирования. Перемежающиеся неисправности возникают случайным образом и моделируются с помощью вероятностных методов (Марковские модели).

4. Генерация тестов

Непрерывное совершенствование технологий проектирования и производства цифровых изделий приводит к увеличению плотности компоновки и сложности устройств, достижение необходимого уровня надежности которых обеспечивается тестированием. Для решения задачи тестирования современных сверхсложных электронных устройств необходимы новые, более эффективные методы построения тестов [47-52]. Производство систем на кристаллах (system-on-chip, SoC) с использованием технологии глубокого субмикрона (Deep Submicron, DSM) позволяет снизить затраты на производство, но при этом стоимость тестирования остается неизменной и представляет собой значительную часть общей стоимости проекта. Уменьшить затраты на тестирование изделия можно путем повторного использования блоков интеллектуальной собственности (IP cores), а также разработки моделей и методов тестирования SoC на высоком уровне иерархии [53]. Высокоуровневые модули системы на кристалле описываются в терминах поведения функциональных компонентов, что не позволяет использовать для их тестирования готовые технические решения вентильного уровня. Классическая модель одиночной константной неисправности (stuck-at fault), представляющая внутренние логические вентили или их межсоединения, не применима для использования на системном уровне. Структурное высокоуровневое тестирование не может быть выполнено с использованием готовых тестовых решений, поскольку генерация теста выполняется после структурного синтеза. Реализация тестирования зависит от технологии изготовления SoC и изменяется в процессе жизненного цикла изделия [36].

Для обеспечения возможности многократного использования тестов в новых проектах необходимо разработать такую модель неисправности, которая является независимой от реализации системы на кристалле. Следует также найти ответы на вопросы: 1) Может ли тест, построенный на базе функциональной модели неисправности, быть эффективно использован для непокрываемых тестом физических дефектов? 2) Как эффективность теста зависит от синтезируемой структуры? Эти вопросы являются важными не только с точки зрения повторного использования тестов, но также из-за того, что программные модули

могут быть синтезированы с помощью существующих систем автоматизированного проектирования и сохранены в библиотеках IP модулей.

Для успешного проектирования и производства изделия необходимы методология тестирования и модели неисправностей, которые обеспечивают высокоуровневую валидацию проекта [36].

Дефект-ориентированное тестирование, основанное на генерации тестов на транзисторном уровне и использовании токовых моделей (IDDQ), эффективно применяется в технологии глубокого субмикрона. IDDQ метод тестирования основан на измерении тока и хорошо работает, когда средний ток схемы с неисправностью больше, чем ток исправного устройства [37]. Дефект-ориентированное тестирование начинается после реализации этапа размещения и трассировки (place and route) (рис. 6) [36].

Сущностью константной модели неисправностей является абстракция реального дефекта. Модель является основой для автоматической генерации тестовых наборов и формирования алгоритмов моделирования неисправностей. Условием достижения высокого уровня покрытия неисправностей является то, что тест должен транспортировать некоторое конкретное значение (значения) в дефектную область от управляемых точек ввода и далее до наблюдаемых выходов в целях обнаружения неисправного поведения. Данная модель наиболее эффективна для тестирования на кристалле (post-silicon testing). Генерация тестов для константных неисправностей выполняется перед размещением и трассировкой проекта (рис. 6). Тесты для константных неисправностей покрывают реальные дефекты топологии только примитивных вентилей. В этом случае говорят о генерации тестов, не зависящих от топологии кристалла.

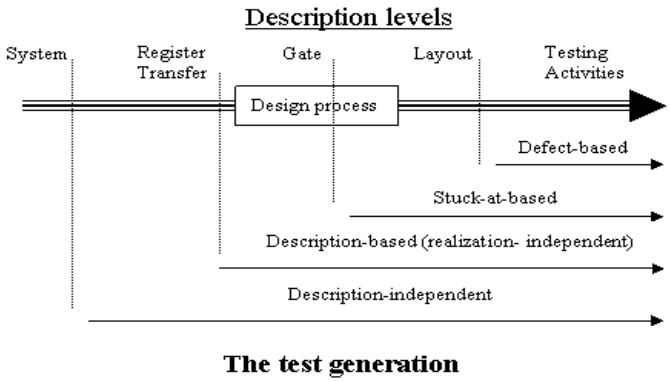

Рис. 6. Генерация тестов в процессе проектирования

Известны несколько подходов к генерации тестовых наборов на уровне регистровых передач (Register Transfer Level, RTL) [36]: использование двоичных деревьев решений, выявление искажений в RTL описании схемы, объединение статического анализа с симуляцией. Большинство из них позволяют генерировать тестовые последовательности удовлетворительного качества, совместимые со средствами ATPG вентильного уровня. Главным преимуществом тестирования устройства на RTL уровне является то, что размерность описания схемы здесь намного меньше, чем на логическом уровне. При генерации тестов на уровне регистровых передач набор тестовых последовательностей создается для всех возможных реализаций, и можно говорить о генерации тестов, не зависящих от имплементации. Задача генерации тестов может выполняться параллельно с синтезом схемы на вентильном уровне (см. верхнюю часть рис. 7).

Генерация тестов на системном уровне зависит от используемой модели неисправностей. В этом случае не только реализация, но и синтезируемое поведенческое описание не известны. Задача генерации тестов в этом случае является более сложной, но может решаться одновременно с формированием синтезируемого описания и синтезом схемы на вентильном уровне (см. рис. 7) [36].

Перед отправкой проекта в производство необходимо сформировать тестовые наборы. Верификация устройства предполагает запуск тестовой программы на рабочей модели кристалла. Тестеры являются дорогостоящими компонентами и могут использоваться в течение длительного времени. Начало разработки тестов в конце процесса проектирования значительно увеличивает время выхода изделия на рынок. Если используется методология проектирования сверху вниз, то системная модель изделия на кристалле формируется в самом начале процесса проектирования и может быть использована при разработке тестовой программы. Таким образом, инженер-тестировщик может стать участником процесса проектирования на ранних стадиях и использовать виртуальный прототип устройства в виде системной модели. Это позволит существенно уменьшить время проектирования и стоимость изделия.

Методология проектирования сверху вниз ориентирована на автоматический синтеза списка соединений вентильного уровня с использованием поведенческого описания или системной модели. Время выхода изделия на рынок (time-to-market) зависит от длительности процедуры логического синтеза, продолжительности тестопригодного проектирования и генерации тестов (см. рис. 7). Тестопригодное проектирование и генерация тестов на основе системной модели позволит сократить время выхода на рынок [36, 45]. Анализ тестового покрытия на вентильном уровне не является времязатратной процедурой и дает возможность уменьшить длину тестовых последовательностей, полученных на системном уровне.

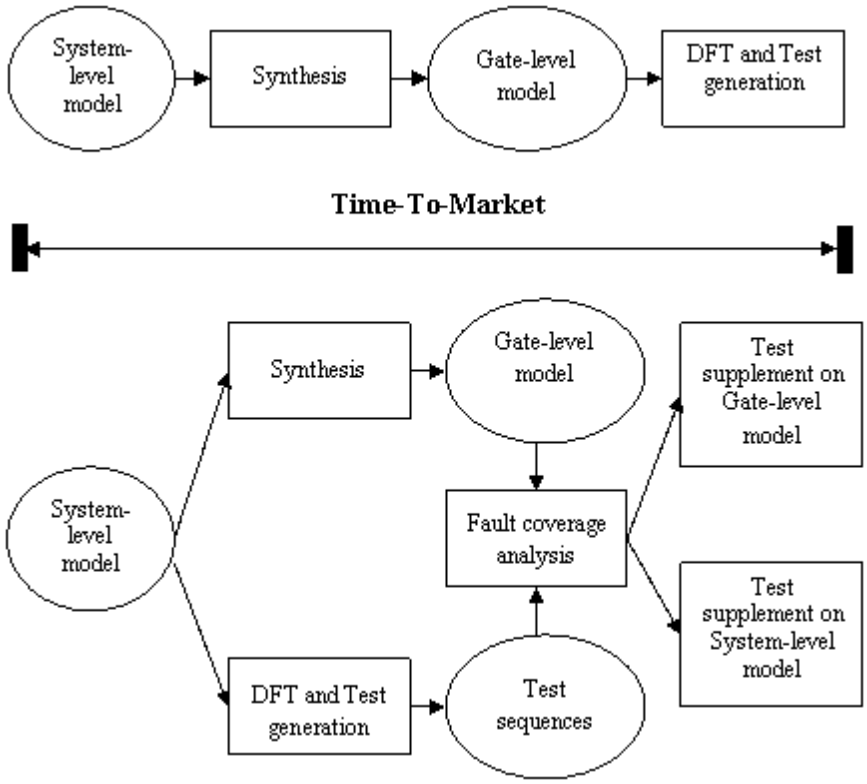

Рис. 7. Время time-to-market на различных этапах процесса проектирования

Генерация тестов на системном уровне не может гарантировать 100% покрытие неисправностей вентильного уровня для каждой возможной имплементации. Формирование тестовых последовательностей необходимо выполнять параллельно на системном и на вентильном уровнях, поскольку вероятность генерации тестов для труднообнаруживаемых неисправностей может быть различной на каждом уровне описания.

5. Модели функциональных неисправностей

Все более широкое использование программно-аппаратных систем в критических приложениях привело к повышению значимости верификации и тестирования программных и аппаратных модулей. В настоящее время существует ряд проблем верификации, связанных с высокой сложностью программно-аппаратных приложений и их гетерогенной структурой [35, 48-53]. Стоимость верификации системы увеличилась до такой степени, что иногда даже превышает стоимость проекта. Формальные методы верификации позволяют проверить функциональность с помощью формальных методов (проверка моделей, проверка эквивалентности, автоматическое доказательство теорем). Для управления сложностью задачи верификации предложены методы, основанные на симуляции (эмуляции) описания системы заданной входной последовательностью.

Функциональные неисправности искажают пространство состояний цифрового изделия, представленное спецификацией. Дефект проектирования представляет собой неправильную деталь проекта, сформированную разработчиком. Дефекты проектирования являются следствием либо синтаксических (семантических) ошибок в описании устройства, либо фундаментального непонимания функциональности, описанной проектной спецификацией. Количество потенциальных дефектов проектирования может быть слишком большим, чтобы с ними можно было бороться автоматически или вручную, поэтому необходимо применять способы уменьшения сложности проекта без ущерба для точности результатов. Модель проектной неисправности описывает поведение некоторого множества дефектов проектирования. Модель функциональной неисправности описывает физические и проектные дефекты аппаратных и программных модулей. Модель функциональной неисправности можно оценить точностью моделирования проектных дефектов и эффективностью (количеством обнаруживаемых неисправностей схемы).

Большинство аппаратных систем разрабатываются на основе методологии проектирования сверху вниз, которая начинается с поведенческого описания системы. Как результат, большинство моделей функциональных неисправностей являются моделями поведенческого или алгоритмического уровня. Существующие модели функциональных неисправностей могут быть классифицированы по стилю поведенческого описания, на котором они базируются. Системное поведение описывается на языках программирования (System C) или описания аппаратуры (VHDL, Verilog) и преобразуется во внутренний формат для использования в процессе симуляции.

5.1. *Текстовые (семантические) модели неисправностей.* Текстовая (семантическая) модель неисправностей применяется для исходного текстового поведенческого описания проекта [38]. Простейшей текстовой моделью является метрика покрытия инструкций (statement coverage metric), используемая при тестировании программного обеспечения, которая связывает потенциальную ошибку с каждой строкой кода и требует, чтобы каждый оператор поведенческого описания выполнялся во время тестирования [38]. Эта модель не очень эффективна, поскольку количество возможных неисправностей равно числу строк кода. Ограничение точности покрытия инструкций позволяет в сочетании с другими моделями неисправностей повысить эффективность тестирования.

Ряд моделей функциональных неисправностей базируются на обходе путей графа потоков управления (Control-Data Flow Graph, CDFG), описывающего поведение системы [38]. Ранние модели неисправностей CDFG основывались на покрытии ветвей и путей графа. Покрытие ветвей предполагает, что множество всех проверяемых путей графа CDFG охватывает два направления реализации всех бинарных условий. Покрытие ветвей широко используется при тестировании программного и аппаратного обеспечения, однако применение только данной модели не позволяет получить полную гарантию корректности кода.

Метрика покрытия путей является более эффективной по сравнению с метрикой покрытия ветвей, поскольку она отражает количество путей графа потоков управления. Предполагается, что дефект связан с некоторым путем графа потоков управления и, следовательно, для гарантированного обнаружения всех неисправностей должны быть выполнены все пути потока управления. Количество путей управления может быть бесконечным, если граф CDFG содержит цикл. Поэтому метрика обхода путей может быть ограничена длиной пути. Поскольку общее число путей потока управления растет экспоненциально с количе-

ством условных операторов, можно выбрать подмножество всех путей потока управления, необходимое и достаточное для тестирования. Одним из критериев выбора пути может быть базисный набор путей или подмножество путей, которые линейно-независимы и могут образовывать любой другой путь. При тестировании потоков данных появление каждой переменной рассматривается либо как описание переменной, либо как ее использование. При выборе пути рассматриваются такие, которые связывают определение переменной с ее применением. Критерии тестирования потоков также применяются для проверки поведенческого описания аппаратных модулей.

Большинство моделей неисправностей графа потоков управления рассматривают пути без ограничения значений переменных и сигналов. В противоположность им существуют модели неисправностей, ориентированные на переменные/сигналы, которые включают более жесткие ограничения на величину сигнала в целях обнаружения неисправностей. Методика анализа домена при тестировании программного обеспечения рассматривает не только путь потока управления, но и значения переменных и сигналов во время выполнения. Домен представляет собой подмножество пространства входных элементов программы, в котором каждый элемент активизирует выполнение программы по некоторому пути. Неисправность домена вызывает выполнение программы, следствием которого является переход в неправильную область. Данный метод может быть также применен для тестирования аппаратного обеспечения.

Многие модели неисправностей графа потоков управления содержат требования к активизации неисправностей независимо от значения наблюдаемости. Для устранения этого недостатка предложены поведенческие модели неисправностей, применимые для тестирования software и hardware модулей. Подход OCCOM [38] базируется на добавлении неисправностей, называемых тегами, к каждому определению переменной, представляющих положительное или отрицательное смещение от правильного значения сигнала. Знак ошибки известен, но величина – нет. Анализ наблюдаемости вдоль пути потока управления делается вероятностно, с использованием алгебраических свойств операций и данных моделирования. Тег будет распространяться с помощью поведенческой операции, если будут выполнены два условия: 1) совпадет знак; 2) другие входы в процессе выполнения операции не контролируются. Разработан также точный метод определения наблюдаемости, в котором константная неисправность вводится на внутренних переменных, и ее распространение обеспечивается поведением объекта. Поскольку анализ наблюдаемости является точным, вычислительная сложность при этом возрастает.

5.2. *Мутационные модели неисправностей.* Мутационное тестирование основано на искусственном внесении неисправностей в код программы и применяется для тестирования программного и аппаратного обеспечений [38]. Основная идея заключается в имитации типичных ошибок программиста и создании специальных тестов для их выявления (тестов, которые бы обнаруживали неисправности, если бы они присутствовали). Неисправности вводятся в оригинальную программу и создается много неисправных версий программы. Каждая из них содержит одну ошибку. Ошибочные программы называются мутантами оригинальной программы. Целью генерации тестов является разграничение оригинальной программы и всех ее мутантов. Оригинальная программа и все ее мутации тестируются на одном и том же наборе тестов. Если на этом наборе подтверждается правильность программы и выявляются все ошибки в программах-мутантах, то оригинальная программа объявляется правильной. Если в некоторых мутантах не были выявлены все ошибки, то тестовый набор считают не полным, и он подлежит доработке. Набор мутационных операторов языка VHDL включает изменения следующих объектов: арифметические операторы, определение и изменение абсолютного значения и константы, логические операторы, реляционные операторы, добавление унарного оператора. Каждый оператор представляет определенный класс неисправностей. Все возможные изменения в программе не могут рассматриваться из-за их непомерно большого количества. Изменения могут быть ограничены до приемлемого набора на основе двух гипотез: эффект сцепления и компетентный программист. Эффект сцепления гласит, что сложные неисправности могут сочетаться с простыми неисправностями, таким образом, тестовый набор, который обнаруживает все простые ошибки в программе, будет обнаруживать также и сложные неисправности.

Гипотеза «компетентный программист» утверждает, что компетентный программист стремится писать программы, которые практически являются правильными. Другими словами, программа, написанная компетентным программистом, может быть неправильной, но она будет отличаться от правильной версии относительно простыми ошибками. Недостатком мутационных моделей является локальный характер мутаций, что ограничивает применение моделей для описания большого набора дефектов проектирования.

5.3. *Модели неисправностей конечного автомата.* Конечные автоматы (КА) являются классическим способом описания поведения последовательностных схем, и для них определены модели неисправностей [38]. Наиболее распространенной является модель покрытия состояний, основанная на требовании покрытия всех возможных состояний и выполнения всех возможных переходов в процессе тестирования. Дефекты КА не выводят его за пространство состояний, заданное спецификацией. Проблема использования моделей неисправностей конечного автомата заключается в высокой сложности решения задачи тестирования, которая обусловлена большой размерностью пространства состояний типовой вычислительной системы. Решением указанной проблемы может быть выявление подмножества состояний конечного автомата, имеющих решающее значение для его корректного функционирования. Модели расширенного конечного автомата (Extended Finite State Machine, EFSM) и машины контроля (Extracted Control Flow Machine, ECFM) позволяют уменьшить конечный автомат путем его разделения на пространство состояний и пространство данных. Уменьшенный конечный автомат генерируется путем проецирования оригинального конечного автомата на множество состояний, имеющих наибольшее значение для процесса валидации.

6. Модель исправного поведения объекта

Модель объекта диагностирования – это совокупность гетерогенных компонентов, взаимосвязанных во времени и пространстве, с заданной адекватностью описывающих некоторый процесс или явление. Модель может быть представлена в аналитической, табличной, векторной, графической или другой форме и задана в явном или неявном виде [28, 29, 32].

Явная модель объекта диагностирования состоит из описаний его исправной и всех неисправных модификаций. Неявная модель содержит описание исправного объекта, модели его физических неисправностей и правила получения по ним все неисправных модификаций объекта. Универсальной математической моделью объекта диагностирования является таблица функций неисправностей (ТФН). Каждое неисправное состояние объекта диагностирования соответствует одной неисправности (одиночной или кратной) из заданного класса неисправностей. Недостатком ТФН являются ее большие размеры. Модель дискретной системы может быть представлена в виде таблицы истинности, логической сети, альтернативного графа, эквивалентной нормальной формы представления булевых функций, таблицы переходов-выходов многотактной схемы. Выбор модели влияет на глубину и трудоемкость процесса диагностирования.

7. Логическое моделирование

Логическое моделирование является формой верификационного тестирования проекта с использованием модели проектируемой системы. На вход модели подаются входные стимулы, выполняется построение и анализ временных диаграмм для внешних входов, выходов схемы и внутренних линий [35].

Верификация проекта позволяет проверить функциональные режимы относительно спецификации. Проверка осуществляется на каждой стадии преобразования модели от системного уровня до уровня имплементации проекта в кристалл путем сравнения результатов, полученных в процессе моделирования, и эталонных результатов, предусмотренных спецификацией.

Задачи, решаемые в процессе логического моделирования: 1. Проверка правильности функционирования цифровой схемы. 2. Исследование временных параметров схемы (быстродействие, время выполнения операций, тактовая частота). Обнаружение состязаний, рисков сбоя, анализ задержек. 3. Оптимизация проектных решений. 4. Генерация временных диаграмм. 5. Генерация тестовых последовательностей. 6. Моделирование неисправностей.

Для верификации проекта необходим прототип устройства, функционирующий на заданной рабочей частоте, однако создание прототипа является дорогостоящим и трудоемким процессом. Замена прототипа программной моделью называется симуляцией. Верификация проекта с использованием симулятора имеет следующие преимущества:

– проверка ошибочных условий (например, конфликты шин);

– возможность изменения задержек в модели для проверки граничных временных параметров;

– проверка задаваемых пользователем значений параметров схемы;

– возможность начала моделирования схемы на любом этапе проектирования;

– точный контроль тайминга асинхронных событий (например, прерываний);

– возможность формирования автоматизированного тестового окружения моделируемой схемы, использование RTL модели для управления и наблюдения за поведением схемы в процессе моделирования.

Технологии использования аппаратного прототипирования с помощью PLD имеют ряд существенных преимуществ по сравнению с программным симулятором: быстродействие и возможность коррекции проекта.

Методы логического моделирования можно классифицировать по следующим признакам: 1) в зависимости от способа учета времени распространения сигнала – синхронные (без учета задержек в элементах схемы) и асинхронные (с учетом задержек); 2) в зависимости от способа представления сигналов – двоичные и многозначные (троичные, пятизначные); 3) по способу организации работы программы –компилятивные и интерпретативные; 4) в зависимости от организации очередности моделирования – пошаговые и событийные.

Синхронное моделирование предназначено для анализа переходных процессов в цифровых устройствах вентильного и функционального уровней описания на основе моделей элементов, представленных их логическими функциями без учета задержек сигналов. В процессе моделирования вычисляют значения сигналов на выходах логических элементов схемы по заданным входным сигналам. При этом предполагается, что время существования переходного процесса намного больше номинальной задержки схемы. Синхронное моделирование наиболее эффективно используется для анализа работы комбинационных схем в установившемся режиме. Результат моделирования в этом случае наиболее точно соответствует реальному режиму работы устройства. К методам синхронного моделирования относят:

– метод Эйхельбергера, предназначенный для синхронного анализа переходных процессов в цифровых устройствах вентильного уровня описания;

– многозначное синхронное моделирование, позволяющее обнаруживать все реальные состязания в схеме. Этот метод иногда указывает на ложные состязания, что приводит к дополнительным затратам при логической верификации цифровых систем.

Решением указанной выше проблемы являются асинхронные методы анализа цифровых схем. Их разнообразие определяется значностью алфавита моделирования и степенью адекватности моделей по реальным временным параметрам.

Асинхронное моделирование применяется для анализа переходных процессов в логических схемах с учетом времени распространения сигналов в элементах и соединительных цепях схемы. Каждый компонент схемы характеризуется некоторой средней задержкой, значение которой может изменяться в зависимости от режима работы компонента, комбинации входных сигналов, температуры, отклонений в технологии изготовления.

К методам асинхронного моделирования относят:

– Δ-троичное моделирование, которое устраняет недостатки двоичного асинхронного и троичного синхронного методов;

– асинхронное троичное моделирование с нарастающей неопределенностью, которое устраняет детерминизм в модельной задержке компонента схемы, заключая ее в некоторый интервал.

8. Моделирование неисправностей

Сущность моделирования неисправностей состоит в определении влияния одного или нескольких дефектов на состояния линий объекта при подаче тестовых последовательностей [39, 54]. Методы моделирования неисправностей можно классифицировать следующим образом: одиночное, параллельное, дедуктивное, кубическое и совместное моделирование.

Одиночное моделирование неисправностей базируется на внесении одной одиночной константной неисправности эквипотенциальной линии в схему. При подаче тестовых последовательностей выполняется анализ проявления неисправности на внешних выходах объекта диагностирования. Метод ориентирован на обработку схем нерегистрового уровня и не требует значительных временных затрат.

Параллельное моделирование неисправностей основывается на использовании машинных команд параллельной обработки слов (регистров): логическое сложение, умножение, инверсия, исключающее ИЛИ. Метод относится к компилятивному моделированию, поскольку поведение примитивов схемы описывается с помощью алгоритмических языков или ассемблеров. В процессе моделирования одновременно выполняется анализ P неисправностей на входном наборе, где P – разрядность машинного слова, доступного для параллельной обработки. К недостаткам метода относят сложность проектирования моделей и их ориентацию на конкретную вычислительную платфрму. Быстродействие метода в P раз выше одиночного моделирования неисправностей. Идея параллельной обработки бинарного вектора с помощью только логических операций может быть использована для существенного увеличения скорости моделирования.

Дедуктивное моделирование неисправностей заключается в одновременной обработке всех одиночных константных неисправностей схемы на одном входном наборе и выделении при этом подмножества проверяемых дефектов. Метод ориентирован на вентильный уровень описания модели проектируемого объекта в базисе И-ИЛИ-НЕ. Необходимость получения аналитических формул для каждого типа примитивного элемента и большие затраты памяти для хранения списков неисправностей усложняют практическую реализацию метода.

В совместном (конкурентном) моделировании, как и в дедуктивном, выявляются сразу все проверяемые неисправности для данного входного набора. Метод ориентирован на обработку различных типов моделей схем, неисправностей, задержек и сигналов. В отличие от дедуктивного метода, где дефекты моделируются неявно, конкурентный алгоритм анализирует явно исправную работу и те неисправности, которые модифицируют состояния входов или выходов схемы, что используют эффективные модели элементов, такие как табличные и функциональные.

Дедуктивно-параллельное моделирование неисправностей цифровых систем основывается на применении преимуществ дедуктивного и параллельного алгоритмов [39] и позволяет за одну итерацию обработки моделируемой схемы выявить все неисправности, проверяемые на тест-векторе. Метод позволяет существенно повысить быстродействие моделирования одиночных константных неисправностей для оценки качества синтезируемых тестов цифровых систем, имплементируемых в ПЛИС, содержащих миллионы вентилей.

9. Методы диагностирования неисправностей

Задачами технического диагностирования являются: определение технического состояния объекта, поиск места и определение причин отказа. Техническое состояние объекта определяется с помощью специальной тестовой последовательности входных воздействий. Методы формирования тестовых последовательностей для диагностирования неисправностей можно условно разделить на несколько типов.

1. Разделение, использование деревьев решений – заключается в моделировании поведения исправной системы и систем с N заранее определенными неисправностями. Отклик каждой из них на входное воздействие применяется для формирования (N+1) систем. При этом должны выполняться условия:

– исправная система должна быть быстро отделена от неисправных (выявление неисправностей);

– все системы являются однозначно идентифицируемыми (различимыми) (выявление неисправностей и определение их местоположения).

Результирующее разделение или построение дерева решений определяет диагностическую тестовую последовательность, которая позволяет однозначно определить принадлежность тестируемой системы одной из (N+1) категорий.

Для поиска неисправности применяют последовательный, комбинационный и последовательно-комбинационный методы. Последовательный метод заключается в таком построении процедуры поиска неисправностей, при котором информация о состоянии отдельных тестируемых систем вводится и логически обрабатывается последовательно. Реализация метода заключается в основном в определении очередности контроля. Программа поиска при этом может быть жесткой или гибкой. Жесткая программа предусматривает наличие заранее определенной последовательности контроля. При гибкой программе содержание и порядок последующих проверок зависят от предыдущих результатов. Комбинационный метод заключается в том, что на вход тестируемой системы подается фиксированный набор тестов. Диагноз формируется только после того, как будут получены отклики на все тестовые воздействия.

2. Активизация одномерного пути. Данный метод основывается на введении известной неисправности в схему и ее транспортировании на один из первичных выходов по активизированному пути. При этом любое изменение логического значения в месте неисправности приводит к изменению значения на соответствующем первичном выходе. Описанная процедура носит название прямой фазы. Обратная фаза заключается в определении значений других первичных входов и выходов, таких, чтобы заданная неисправность проявлялась на первичном выходе. Метод прост и удобен в использовании, однако в схеме могут существовать неисправности, для проверки которых необходимо активизировать несколько путей (в случае наличия сходящихся разветвлений).

3. Использование таблицы функций неисправностей и таблицы неисправностей. Таблица функций неисправностей является специальной формой представления поведения объекта диагностирования в исправном и неисправном состояниях. Таблица неисправностей связывает набор тестов и проверяемые ими неисправности. Ограничением данного метода является размерность указанных таблиц.

4. Метод булевых производных. Булева производная определяется путем выполнения операции OR над двумя булевыми функциями, представляющими исправный и неисправный объект. Если булева производная равна 1, считается, что проявляется ошибка и определяется соответствующая тестовая последовательность. Тестовые наборы определяются путем формирования булевой производной для каждой неисправности.

5. Метод эквивалентной нормальной формы основан на представлении булевой функции в виде эквивалентной нормальной формы, описывающей конкретную реализацию схемы. Эквивалентная нормальная форма может быть вычислена методом подстановки, с той разницей, что избыточные термы не исключаются, так как они характеризуют конкретную реализацию схемы.

Двоичное моделирование или эмуляция квантовых схем на классических компьютерах не отражает всех свойств квантовых схем. Эмуляция квантовых компонентов на классических компьютерах не дает существенных результатов по увеличению быстродействия [21-25]. В то же время моделирование цифровых классических схем на основе использования отдельных свойств кубита может обогатить «бедные» алгоритмы классических машин. Например, существует еще не решенная проблема, к которой можно подступиться: как за один такт промоделировать 2 в степени n входных набора, поданных на цифровую схему, если эти наборы представлены одним двоичным вектором-кубитом?

10. Заключение

Таким образом, анализ современных рыночных технологий синтеза, анализа, тестирования, диагностирования и восстановления работоспособности цифровых систем на кристаллах, описанный в первом разделе, дает основания сформулировать актуальные на рынке электронных технологий тенденции в области Design and Test, сущность котрых заключается в создании кубитных структур данных и «квантовых» методов моделирования, верификации, диагностирования и восстановления работоспособности, интегрированных в инф-

раструктуру сервисного обслуживания программно-аппаратных компонентов цифровых систем на кристаллах в целях повышения качества изделий или выхода годной продукции на основе уменьшения времени ремонта за счет увеличения аппаратной стоимости вычислительных проектов.

Основная инновационная идея квантовых или кубитных вычислений заключается в переходе от вычислительных процедур над байт-операндом, определяющим в дискретном пространстве одно точечное решение, к квантовым параллельным процессам над кубит-операндом, одновременно формирующим булеан решений. Функция цели есть повышение качества цифровых изделий за счет создания инфраструктуры, включающей технологии диагностирования и ремонта дефектных компонентов цифровой системы на кристалле L, минимизация времени восстановления работоспособности T за счет программно-аппаратной избыточности H инфраструктуры SoC:

$$E = F(L,T,H) = \min[\frac{1}{3}(L + T + H)],$$
$$Y = (1-P)^n;$$
$$L = 1 - Y^{(1-k)} = 1 - (1-P)^{n(1-k)};$$
$$T = \frac{(1-k)\times H^s}{H^s + H^a};\ H = \frac{H^a}{H^s + H^a},$$

где L – инверсная переменная выхода годной продукции Y, зависящая от тестопригодности проекта k, вероятности P существования дефектных блоков и числа необнаруженных неисправностей n. Время тестирования и диагностирования зависит от тестопригодности архитектуры k, умноженной на число примитивов, деленное на общее количество логических примитивов проекта. Нефункциональная избыточность для ремонта устройства зависит от сложности инфраструктуры, деленной на аппаратную сложность проекта.

Список литературы: 1. *Дуплий В.А.* Квантовая информация, кубиты и квантовые алгоритмы / С.А. Дуплий, В.В. Калашников, Е.А. Маслов // Вісник Харківського університету. 2005. №657. С. 99-104. **2.** *Нильсен М.* Квантовые вычисления и квантовая информация / М. Нильсен, И. Чанг. Пер. с англ. М.: Мир 2006. 824 с. **3.** *Бекман И.Н.* Лекции по информатике // Электронный ресурс http://www.twirpx.com/file/602601/ **4.** *Benenti G., Casati G., Strini G.* Principles of Quantum Computation and Information. Volume 1: Basic Concepts. World Scientific. 2004. 256 p. **5.** *Vedral V., Plenio M.B.* Basics of Quantum Computation. 1998. 28 p. Электронный ресурс: arXiv:quant-ph/9802065 v1 25 Feb 1998 http://www.tfp.uni-karlsruhe.de/~cuevas/Lehre/SS04/9802065.pdf. **6.** *Rosinger E.E.* Basics of Quantum Computation (Part 1). 2004. 87 p. Электронный ресурс: arXiv:quant-ph/0407064 v1 8 Jul 2004 http://chaos.swarthmore.edu/courses/TSG/2004d.pdf **7.** *Stenholm S., Suominen K.-A.* Quantum approach to informatics. A John Wiley & Sons, Inc., Publication. 2005. 249 p. **8.** *Imai Hiroshi, Hayashi Masahito.* Quantum Computation and Information. From Theory to Experiment. Springer. 2006. 234 p. **9.** *Nielsen M.A., Chuang I.L.* Quantum Computation and Quantum Information. Cambridge University Press. 2010. 710 p. **10.** *Mikio Nakahara.* Quantum computing: an overview // Mathematical Aspects of Quantum Computing. 2007. 53 p. http://www.worldscibooks.com/physics/6851.html **11.** *Whitney M.G.* Practical Fault Tolerance for Quantum Circuits. A dissertation submitted in partial satisfaction of the requirements for the degree of Doctor of Philosophy in Computer Science in the Graduate Division of the University of California, Berkeley. 2009. 206 p. **12.** *DiVincenzo D.P.* The Physical Implementation of Quantum Computation // IBM T.J. Watson Research Center, Yorktown Heights, NY 10598 USA. 9 p. Электронный ресурс: arXiv:quant-ph/0002077 v3 13 Apr 2000. http://www.unifiedfieldtheories.com/0002077_DiVincenzo_Phys_Imp.pdf **13.** *Svore K.M., Terhal B.M., DiVincenzo D.P.* Local Fault-Tolerant Quantum Computation // Электронный ресурс: arXiv:quant-ph/0410047v2 6 Jun 2005. http://research.microsoft.com/pubs/143764/local2005.pdf **14.** *Бейтсон Г.* Шаги в направлении экологии разума / Г. Бейтсон. М.: КомКнига. 2005. 248 с. **15.** *Валиев К.А.* Квантовые компьютеры: надежды и реальность / К.А.Валиев, А.А.Кокин. Ижевск: РХД. 2001. 352 с. **16.** *Feinstein D.D.Y.* Computer-Aided-Design Methods for Emerging Quantum Computing Technologies / D.D.Y. Feinstein. BiblioLabsII. 2011. 184 p. **17.** *Hayes J.P.* Testing for Missing-Gate Faults in Reversible Circuits / John P. Hayes, Ilia Polian, Bernd Becker // Proc. Asian Test Symposium. Taiwain. November, 2004. **18.** *Матюшкин И.В.* Квантовые клеточные автоматы / И.В. Матюшкин // Электронный научный журнал «ИССЛЕДОВАНО В РОССИИ». 2011. С. 367-392. Электронный ресурс http://zhurnal.ape.relarn.ru/articles/2011/029.pdf **19.** *Feinstein D.Y.* Partially Redundant Logic Detection Using Symbolic Equivalence Checking in Reversible and Irreversible Logic Circuits /D.Y. Feinstein, M.A. Thornton, D.M. Miller // Design, Automation and Test in Europe, DATE '08. 2008. P. 1378 –

1381. **20.** *Nayeem N.M.* Online Fault Detection in Reversible Logic / N.M. Nayeem, J.E. Rice // Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT). 2011. P.426-434. **27.** *Golubitsky O.* A Study of Optimal 4-bit Reversible Toffoli Circuits and Their Synthesis / O. Golubitsky, D. Maslov // IEEE Transactions on Computers. 2011. P. 1-14. **28.** *Основы* технической диагностики / Под. ред. П.П.Пархоменко. М.: Энергия. 1976. 460с. **29.** *Пархоменко П.П.* Основы технической диагностики (Оптимизация алгоритмов диагностирования, аппаратурные средства) / П.П. Пархоменко, Е.С. Согомонян. Под ред. П.П. Пархоменко. М.: Энергия. 1981. 320 с. **31.** *Novak O.* Handbook of testing electronic systems / O. Novak, E. Gramatova, R.Ubar. Czech University Publishing House. 2005. 402 p. **32.** *Автоматизация* диагностирования электронных устройств/ Ю.В.Малышенко и др./ Под ред. В.П.Чипулиса. М.: Энергоатомиздат, 1986. 216с. **35.** *Хаханов В.И., Хаханова И.В., Литвинова Е.И., Гузь О.А.* Проектирование и верификация цифровых систем на кристаллах. Verilog & System Verilog: Харьков. Новое слово. 2010. 528с. **36.** Электронный ресурс: http://www.scrigroup.com/limba/engleza/92/The-Design-Flow-and-Fault-Mode51775.php **37.** *Андрюхин А.И.* Параллельная генерация тестов для МОП-структур на переключательном уровне / А.И. Андрюхин // Наукові праці ДонНТУ. Серія "Інформатика, кібернетика та обчислювальна техніка". 2010. Вип. 11(164). С. 75-78. **38.** *Harris I.G.* Hardware-Software Covalidation: Fault Models and Test Generation / Ian G. Harris // Design and Test of Computers. Vol. 20, Num. 4. July-August 2003. 12 p. Электронный ресурс: http://www.ics.uci.edu/~harris/pubdir/hldvt01hwsw.pdf **39.** *Семенец В.В., Хаханова И.В., Хаханов В.И.* Проектирование цифровых систем с использованием языка VHDL. Харьков: ХНУРЭ. 2003. 492 с. **40.** *Надежность* технических систем / Под ред. И.А.Ушакова. М.: 1985. 512 с. **41.** *Stanisavljevi M.* Reliability of Nanoscale Circuits and Systems / M. Stanisavljevi, M. Schmid, Y. Leblebici. Springer. 2011. 240 p. **42.** *Fan X.* Fault diagnosis of VLSI designs: cell internal faults and volume diagnosis throughput / Xiaoxin Fan // PhD (Doctor of Philosophy) thesis, University of Iowa. 2012. 134 p. **43.** *Pomeranz I.* Transition Path Delay Faults: A New Path Delay Fault Model for Small and Large Delay Defects / I. Pomeranz, S.M. Reddy // IEEE Transactions on Very Large Scale Integration (VLSI) Systems. 2008. Vol.16, No.1. P. 98–107. **44.** *Pomeranz I.* Selection of a Fault Model for Fault Diagnosis Based on Unique Responses / I. Pomeranz, S.M. Reddy // IEEE Transactions on Very Large Scale Integration (VLSI) Systems. – 2010.– Vol.18, No.11. P. 1533-1543. **45.** *Vanitha K.* Implementation of an integrated FPGA based automatic test equipment and test generation for digital circuits / K. Vanitha, C.A. Sathiya Moorthy // Intern. Conf. on Information Communication and Embedded Systems (ICICES). 2013. P. 741-746. **46.** *Niemann H.* Fault tolerant control based on active fault diagnosis / H. Niemann // Proceedings of the 2005 American Control Conference. 2005.Vol. 3. P. 2224-2229. **47.** *Abramovici M.* BIST-Based Delay-Fault Testing in FPGAs / Miron Abramovici, Charles E. Stroud // Journal of Electronic Testing: Theory & Applications. 2003. Vol. 19, No. 5. P. 549-558. **48.** *Ulbricht M.* A new hierarchical built-in self-test with on-chip diagnosis for VLIW processors / Markus Ulbricht, Mario Scholzel, Tobias Koal, Heinrich Theodor Vierhaus // Design and Diagnostics of Electronic Circuits & Systems (DDECS). April 2011. P.143-146. **49.** *Elm M.* BISD: Scan-based Built-In self-diagnosis / M. Elm, H.-J. Wunderlich // Design, Automation & Test in Europe Conference & Exhibition (DATE). March 2010. P. 1243-1248. **50.** *Huang Yu-Jen.* A built-in self-test scheme for the post-bond test of TSVs in 3D ICs / Yu-Jen Huang, Jin-Fu Li, Ji-Jan Chen, Ding-Ming Kwai, Yung-Fa Chou, Cheng-Wen Wu // VLSI Test Symposium (VTS). May 2011. P.20-25. **51.** *Shianling Wu.* Logic BIST Architecture Using Staggered Launch-on-Shift for Testing Designs Containing Asynchronous Clock Domains / Wu Shianling, Laung-Terng Wang, Yu Lizhen, H. Furukawa, Wen Xiaoqing, W.-B. Jone, N.A. Touba, Zhao Feifei, Liu Jinsong, Hao-Jan Chao, Li Fangfang, Jiang Zhigang // Defect and Fault Tolerance in VLSI Systems (DFT). Oct. 2010. P. 358-366. **52.** *Da Silva F.* The Core Test Wrapper Handbook. Rationale and Application of IEEE Std. 1500™ / F. Da Silva, T. McLaurin, T. Waayers. Springer. 2006. XXIX. 276 p. **53.** *Marinissen E.J.* Guest Editors' Introduction: The Status of IEEE Std 1500 / E.J. Marinissen, Yervant Zorian // IEEE Design & Test of Computers. 2009. No26(1). P.6-7. **54.** *Elm M.* Scan Chain Organization for Embedded Diagnosis / M. Elm, H.-J. Wunderlich // Design, Automation and Test in Europe, DATE '08. 2008. P. 468–473.

Поступила в редколлегию 11.03.2014

Baghdadi Ammar Awni Abbas, аспирант кафедры АПВТ ХНУРЭ. Научные интересы: проектирование и тестирование вычислительных систем. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. +380 57 70-21-326. E-mail: hahanov@kture.kharkov.ua.

Хаханов Владимир Иванович, декан факультета КИУ ХНУРЭ, д-р техн. наук, профессор кафедры АПВТ ХНУРЭ. IEEE Senior Member. IEEE Computer Society Golden Core Member. Научные интересы: проектирование и тестирование вычислительных систем, сетей и программных продуктов. Увлечения: баскетбол, футбол, теннис, горные лыжи. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. +380 57 70-21-326. E-mail: hahanov@kture.kharkov.ua.

Литвинова Евгения Ивановна, IEEE Member, д-р техн. наук, профессор кафедры АПВТ ХНУРЭ. Научные интересы: проектирование и тестирование цифровых систем и сетей на кристаллах. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. +380 57 70-21-421. E-mail: kiu@kture.kharkov.ua.